Conceptual Library

Spencer Damiano

The Conceptual Library is the collect and utilization of the most effective study and focus tools. The hope in this is to help user's build the habit of discipline and break the habit procrastination. The goal is structure that encourages a work flow which uses our natural dopaminergic responses without hijacking them for profit.

This project will be implemented as a modern web application built with a React frontend for responsive user interaction, backed by a Flask REST API that handles business logic and data management. MongoDB serves as our NoSQL database solution, offering flexible data modeling for user profiles, study sessions, and task management. The core functionality centers around three key features: a customizable Pomodoro timer for structured work intervals, a focused task management system, and a minimalist study environment with ambient sound options.

Section 1: Purpose of Project

This project has two overarching purposes.

1. Filling a personal need

   This project is designed after talking to classmates and friends about study tips that I've learned through podcasts and books. It occurred to me that while thins tools tend to only be used in isolation and through means that usually are not interconnected.

   This app is the combination of the tools in a way to sustainably build and maintain proper study habits.

   1.1. Timer – This was originally going to be your standard countdown timer but after trying to use this I decided that my flowmodoro timer would be the most ideal. This timer counts up instead of down with a break being $1/5^{th}$ of time working. The reason for this type of timer is it is easier to see if your ability to focus is improving over time.
   1.2. Tasks and Distraction Lists – These are my tools for directing a flow state. A to do list is a way to direct your larger tasks into manage pieces encouraging a steady flow of dopamine to keep you focused and feel that time is flying by. The distraction list is so once you are in that flow state you don't ruin it by checking on whatever impulse hits you. Instead, you write it down so when you are on a break you can indulge it when it won't harm your focus.
   1.3. Music and ambient soundscapes – Music and background noises have their use in encouraging a good work flow, however are more misused then properly used. The more stimulus you require to maintain your work, the harder it becomes to do over time as your brain gets used to a higher and higher dopamine hit. The plan is if these tools are used that they slowly fade as you enter into a flow state. This way they are only used when needed.
   1.4. Study Breaks – Any sustained amount of work, even mental work, requires breaks. The problem is that we either don't take them, or take too much of them. Breaks will be rewarded on a $1/5^{th}$ proportion. During these breaks, users will be able to access study games and meditations to help reward students for good habits.
   1.5. Study habit tracker – These is just a visualization of all your study habits. This keeps the user accountable of what their study habits actually look like. Most students tend to over estimate the amount of work that they do. Hopefully by tracking their goals and history we can also determine what tools are best for effective use.

2. Feeling a professional need

   This project is also about gaining experience and confidence in my ability to develop software in the AI era. With the rise of AI and it's effectiveness in developing code I realized that not only do I have to learn how to use this tool, I need to learn how to use it to make me a better programmer. In my internship earlier this summer I realized that LLM will get to the point where everyone will be able to be a full stack developer, architect, and DevOpps manager. Some good prompt engineering can make one developer feel like a team. Creating my study app isn't enough to get the attention of these companies. I also need to be learning the entire stack to be competitive.

The project will follow Google's style guides for both Python and JavaScript/TypeScript to maintain consistent code quality. API documentation will use Swagger/OpenAPI specification, and code documentation will use JSDoc for JavaScript/TypeScript and appropriate docstring formats for Python. These standards were chosen because they are well-documented, widely used in the industry, and have good tooling support for students learning best practices.

The application's design will focus on providing a clean, usable interface that works well on standard laptop screens (minimum 1366x768 resolution). While the application may work on smaller devices, the primary focus is on creating an effective study environment for laptop and desktop users, as these are the most common devices for focused study sessions.

The project will follow standard software development lifecycle (SDLC) methodologies, specifically using an Agile approach with iterative development. This will allow for continuous learning and improvement throughout the development process, with regular feedback incorporation from potential users (fellow students) to enhance the application's effectiveness.

After researching industry standards for similar applications, we've identified the following key metrics to ensure quality:

- Page load time < 3 seconds
- Smooth timer operation (no lag or skips)
- Successfully save/load user data
- All features accessible through keyboard navigation
- Error-free operation of core timer and task management features

As a student project, the primary focus will be on learning and implementing basic security practices while being aware of potential legal considerations:

Security:

- Learning and implementing proper password hashing
- Understanding and preventing basic SQL injection attacks
- Implementing user authentication and session management
- Learning about XSS prevention in React applications

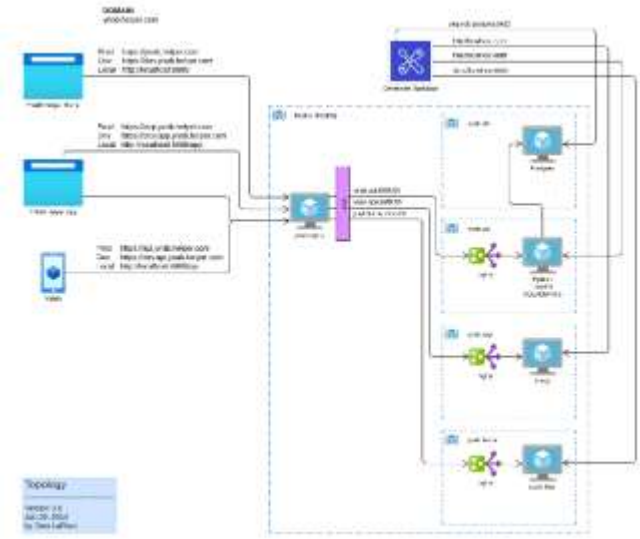
Legal Considerations:

- Research and implement proper licensing for audio content, ensuring all music and sounds are properly licensed for web application use (not just content creation)
- Understanding basic user data protection requirements
- Implementing clear terms of service and privacy policy
- Ensuring all third-party libraries and resources are properly attributed

The application's hosting infrastructure leverages MongoDB Atlas's student program, utilizing the provided $50 in free credits. The development environment operates on an M0 Free Tier Cluster, with the capability to scale to an M2/M5 Shared Cluster for production if necessary. T application will operate well within the free tier limits, making the hosting solution cost-effective for development and initial deployment.

The application will be hosted at conceptual-library.app, with the domain registrar to be selected based on optimal pricing and service reliability. The estimated annual cost for domain registration is approximately $12-15. The total annual operating cost is expected to remain minimal, with MongoDB Atlas covered by student credits and only the domain registration requiring payment. This brings the total annual cost to approximately $15, making the project sustainable on a student budget while maintaining professional quality standards.

The architecture is based off of this topology from an internship that I did.



Section 2a – Must Have Requirements

1) Flowmodoro Timer
    a) Verification Criteria
        i) Users must be able start a study session via the stop watch which caps out at 90 minutes
        ii) Users must be able to initiate a study break timer will be $1/5^{th}$ of the their study time
        iii) Users must be able to have a long study break equal to twice their average study time
        iv) Users must have data stored within the database
    b) Unit tests
        i) A study breaks:
            (1) 90 minutes (test max)
            (2) 5 minutes (potential minimum)
            (3) 10 minutes
            (4) 25 minutes
        ii) Break times
            (1) 18 minutes
            (2) 1 minute
            (3) 2 minute
            (4) 14 minutes (twice the average of the other study breaks)
        iii) Database
            (1) Each of these study breaks are recorded in the database
2) To Do and Distraction lists
    a) Verification Criteria
        i) Users can create, edit, and delete tasks in the todo list
        ii) Users may freely edit, update, and interact with distraction list during any time
        iii) Users must have their tasks persist between sessions in the database
    b) Unit Tests
        i) Todo tests
            (1) Create new task
            (2) Edit existing task
            (3) Delete existing task
            (4) Mark tasks complete
        ii) Distraction
            (1) Add distraction during study time
            (2) Add distraction during break time
            (3) Verify list is add-only during study time
            (4) Verify full access in breaks
        iii) Database
            (1) Verify persistence
            (2) Verify status updates are saved
3) Distraction lists
    a) Verification Criteria

i) Users can create, edit, and delete tasks in the distraction list
ii) Users can add items to the distraction list during study time
iii) Users are not able to view with the distraction list is locked during study periods and only can add to the list
iv) Users may freely edit, update, and interact with distraction list during breaks
v) Users must have their tasks persist between sessions in the database

b) Unit Tests
i) Todo tests
(1) Create new task
(2) Edit existing task
(3) Delete existing task
(4) Mark tasks complete
ii) Distraction
(1) Add distraction during study time
(2) Add distraction during break time
(3) Verify list is add-only during study time
(4) Verify full access in breaks
iii) Database
(1) Verify persistence
(2) Verify status updates are saved

4) User Registration, Authentication, and Management
a) Verification Criteria
i) Users must be able to create new accounts
ii) Users must be able to securely log in to existing accounts
iii) Users must be able to access to only their data
iv) User must be able to save preferences and settings persist between sessions

b) Unit Tests
i) Account Creation:
(1) Create new account with valid credentials
(2) Verify duplicate email prevention
(3) Verify password security requirements
ii) Authentication:
(1) Successful login with correct credentials
(2) Failed login with incorrect credentials
(3) Password reset functionality

5) Users Student Session Management
a) Verification Criteria
i) Study sessions are associated with specific users
ii) Users must be able to switch between study and break times as they define
iii) Users must be able to set minimum study time
iv) Task lists are associated with specific users
v) Users can access their historical study data

b) Unit Tests
i) Data Association:

(1) Verify study sessions link to correct user
(2) Verify task lists link to correct user
(3) Verify data persistence between sessions
(4) Verify data from the dashboard page

6) Music and Soundscapes
   a) Users can enhance their focus with customizable ambient sounds and music that naturally fade as they enter deeper concentration. This prevents dependency on external stimuli while supporting the initial transition into focused study.
   b) Verification Criteria
      i) Users must be able to use playback controls (play, pause, volume) a variety of ambient sounds (white noise, rain, hz beats etc.)
      ii) Users must be able to enable the dopamine protecting where audio fades as the study session persists and audio restarts when a new session is created
      iii) Users must be able to play multiple sound options can be active simultaneously
   c) Unit Tests
      i) Playback functionality
         (1) Start/stop individual sounds
         (2) Adjust volumes levels
         (3) Verify multiple sounds mixing
      ii) Session integration
         (1) Music genre and soundscapes saved
         (2) Smooth audio fading
         (3) Restarted during new session
      iii) Library
         (1) Verify all sounds load
         (2) Selected by genre or scene
         (3) Verify audio quality maintenance

Section 2b - Stretch Requirements

1) Enhanced Analytics

   a) Verification Criteria
      i) Users should be able to see
         (1) Visual representation of study session data over time
         (2) Focus time trends analysis with daily/weekly/monthly views
         (3) Break time utilization metrics
         (4) Task completion correlation with study patterns
      ii) Users should be able to export personal for their own data analysis
   b) Unit Tests
      i) Data Visualization:
         (1) Verify accurate chart generation from study data
         (2) Test different time period selections
         (3) Verify metric calculations accuracy

ii) Trend Analysis:
(1) Test pattern recognition algorithms
(2) Verify statistical calculations
(3) Test recommendation generation
iii) Data Handling:
(1) Verify data aggregation accuracy
(2) Test export functionality
(3) Verify data integrity across views

2) Group Study

a) Verification Criteria
i) Users should create and join study groups
ii) Users should be able to be synchronized with other users in a group study session
iii) Users should be able to see group performance through shared analytics dashboard
iv) Users should be able to participate in group break activity coordination
v) Users should be able to compare metrics between individual and group sessions
b) Unit Tests
i) Group Management:
(1) Create new study group
(2) Add/remove group members
(3) Test group session synchronization
ii) Group Analytics:
(1) Verify shared dashboard updates
(2) Test group vs individual performance metrics
(3) Verify group activity tracking
iii) Session Coordination:
(1) Test group break timing
(2) Verify member status updates
(3) Test group session persistence

3) Study Break Meditations and Games
a) Verification Criteria
i) Users should be able to interact with the break activity menu when in break periods
ii) Users should be able to use s minimum one example of three activity types (game, stretching, mindfulness meditation)
b) Unit Tests
i) Activity Functionality:
(1) Verify activities only accessible during breaks
(2) Test start/pause/skip controls
(3) Verify activity timer synchronization with break timer ii)
ii) Activity Content:
(1) Test all activity types load properly
(2) Verify activity instructions are clear and complete
(3) Test activity progression/completion
iii) Data Tracking:
(1) Verify activity completion records in session data
(2) Test activity preference persistence
(3) Verify proper activity time logging

4) Claude API Integration
    a) Verification Criteria
        i) Users should be able to generate quizzes from user notes
        ii) Users should be able to receive personalized study recommendations based on analytics
        iii) Users should be able to create summary reviews of study sessions
        iv) Users should be able to generate distraction summery from ai
    b) c) Unit Tests
        i) Content Generation:
            (1) Test quiz generation from notes
            (2) Verify summary quality
            (3) Test recommendation relevance
        ii) Integration Testing:
            (1) Verify API response handling
            (2) Test data processing accuracy
            (3) Verify prompt construction
        iii) Performance Analysis:
            (1) Test recommendation effectiveness
            (2) Verify group matching accuracy
            (3) Test system response times

Section 3 – Product overview

Work Flow:

The Conceptual Library application presents users with a comprehensive dashboard interface that serves as their primary workspace. Upon accessing the application, users encounter a centrally positioned Flowmodoro timer, surrounded by a todo list (bottom-left), distraction list (bottom-right), and music controller (bottom-center) - all immediately usable without authentication. When an unauthenticated user starts the timer, a modal appears offering the option to log in for data persistence and tracking, though users may continue without an account. During study sessions, the timer counts upward while users manage tasks via the todo list and record potential distractions in a list that remains view-restricted until break periods. The audio system provides genre-based music selections (randomized to prevent dependency) and basic soundscapes like river or rain sounds, with volume controls that allow mixing of multiple audio sources, all gradually fading as the session progresses. When a study session concludes, unauthenticated users receive a prompt to create an account or log in to save their session data; if they decline, the data is discarded. For authenticated users, the system automatically transitions to a break period (calculated as one-fifth of study time), during which users can access their full distraction list and engage with break activities including games, stretching exercises, and mindfulness meditations. All authenticated user actions - including study sessions, task management, audio preferences, and break activities - are stored in the database and accessible through a profile page that displays pending tasks, recorded distractions, and comprehensive study metrics, creating a seamless, data-driven experience that maintains consistency across multiple study sessions while supporting users who prefer to remain anonymous.

Resources

The Conceptual Library application combines a React with TypeScript frontend and Flask REST API backend, utilizing MongoDB for data persistence. The frontend implements React Context API for state management and incorporates audio processing capabilities through Web Audio API and Howler.js for soundscape mixing. The user interface leverages Tailwind CSS for styling, with specialized components for timer functionality, task management, and audio controls. The backend architecture, built on Flask, handles authentication through JWT, manages WebSocket connections for real-time features (for potential group study stretch goal), and orchestrates data flow through MongoDB collections for users, tasks, sessions, tags, audio preferences, and study statistics. Development tools include Vite, ESLint, and comprehensive testing suites, while deployment utilizes Docker containers and Nginx for hosting. The infrastructure includes monitoring through Sentry, custom analytics, and robust security measures including rate limiting and input validation.

Key Resources and Documentation:

Frontend:

- React: https://react.dev
- TypeScript: https://www.typescriptlang.org
- React Query: https://tanstack.com/query/latest
- Tailwind CSS: https://tailwindcss.com
- Material-UI: https://mui.com
- Chakra UI: https://chakra-ui.com
- Web Audio API: https://developer.mozilla.org/en-US/docs/Web/API/Web_Audio_API
- Howler.js: https://howlerjs.com
- React Timer Hook: https://github.com/amrlabib/react-timer-hook

Backend:

- Flask: https://flask.palletsprojects.com
- MongoDB: https://www.mongodb.com
- MongoDB Atlas: https://www.mongodb.com/atlas/database
- WebSocket Guide: https://developer.mozilla.org/en-US/docs/Web/API/WebSocket

Development Tools:

- Vite: https://vitejs.dev
- Jest: https://jestjs.io
- React Testing Library: https://testing-library.com/docs/react-testing-library/intro
- Pytest: https://docs.pytest.org
- Cypress: https://www.cypress.io

Infrastructure:

- Docker: https://www.docker.com
- Nginx: https://nginx.org
- Sentry: https://sentry.io

Data at Rest

The application uses MongoDB as its primary database system. User information and application data are stored in separate collections within the same database:

Core Collections:

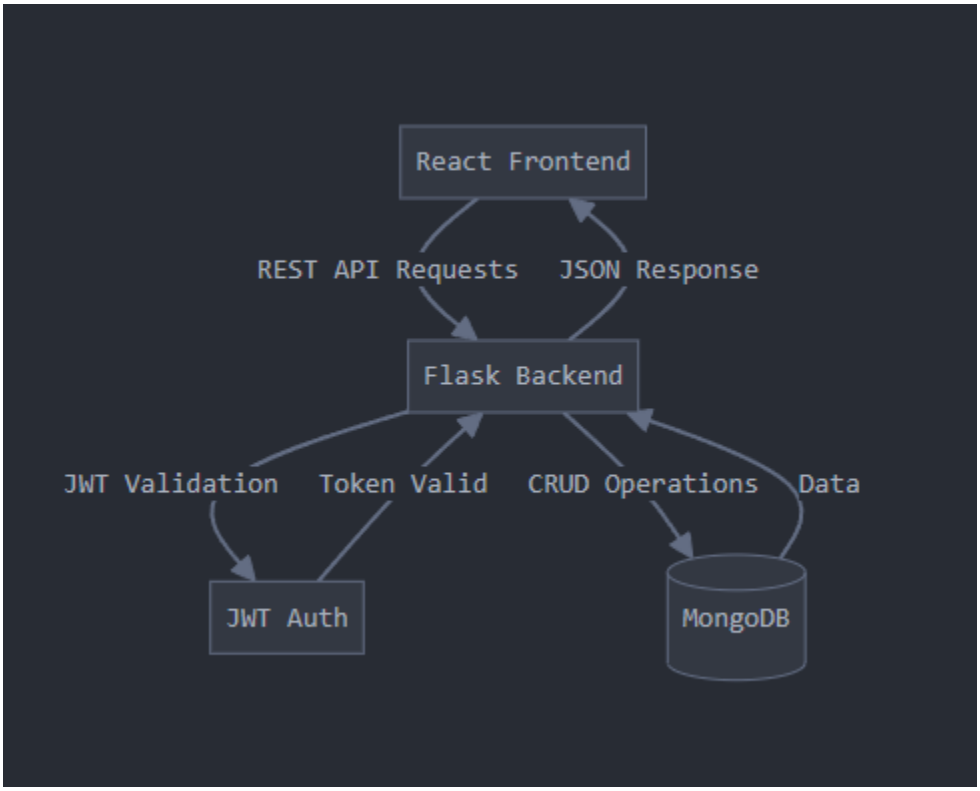
- users: Stores user profiles, authentication info, and preferences
- tasks: Manages todo and distraction list items
- sessions: Records study session data and timestamps
- tags: Stores task categorization data

Data on the Wire

Data transmission uses a simple client-server architecture:

1. Frontend React application communicates with Flask backend via REST API
2. All requests are authenticated using JWT tokens
3. Data is transmitted as JSON over HTTPS
4. Study sessions are tracked through standard API endpoints
5. Audio files are served as static assets

Data State and Flow



Prototype Design

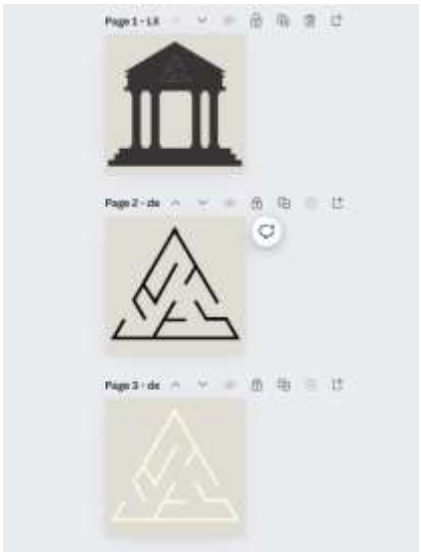


Figure 1: Logos and design

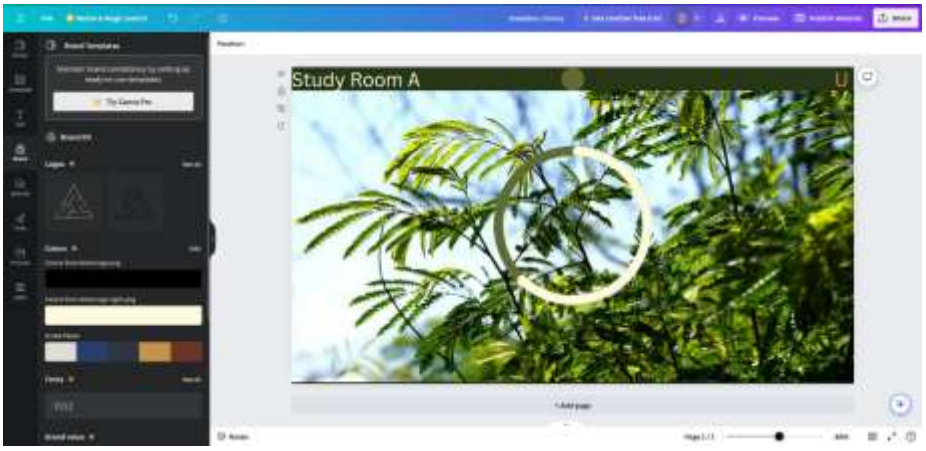


Figure 2: Original Wireframe

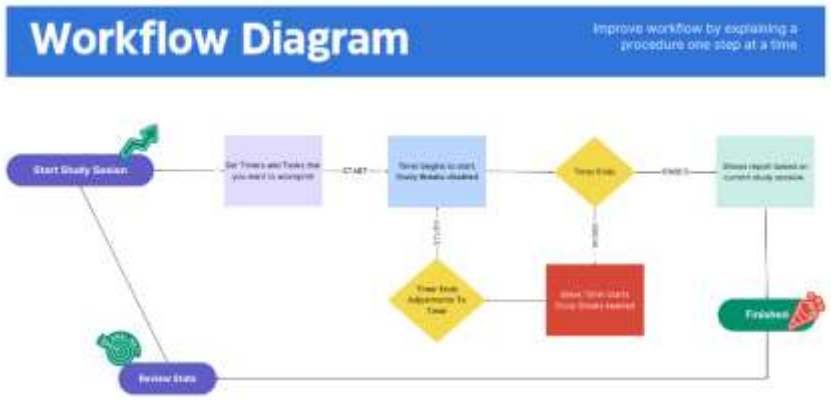


Figure 3: Early Workflow Diagram

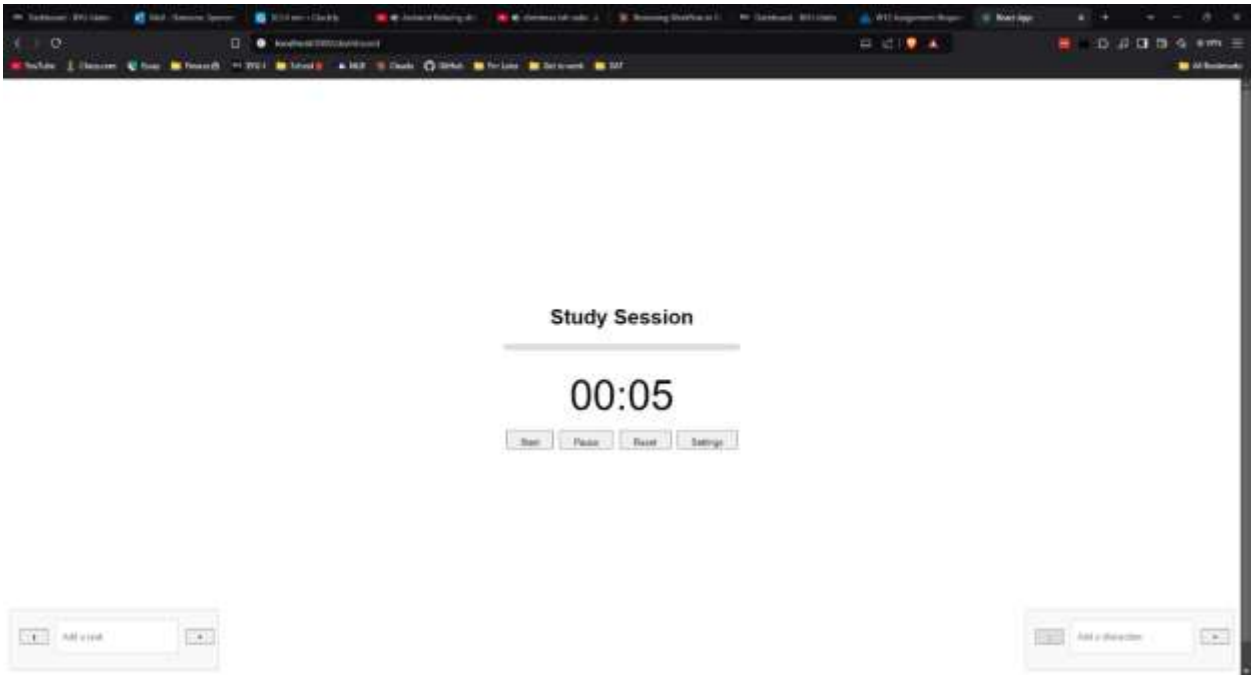


Figure 4: Current Prototype Build